# LAB-11: Image Analysis in Frequency Domain

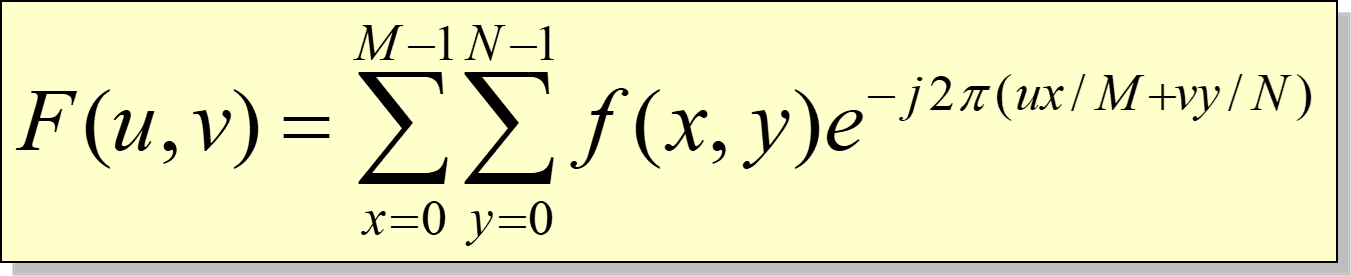
## Objective:

The objective of this lab is to understand Fourier Transform, apply it on images and understand the results.

## Theory:

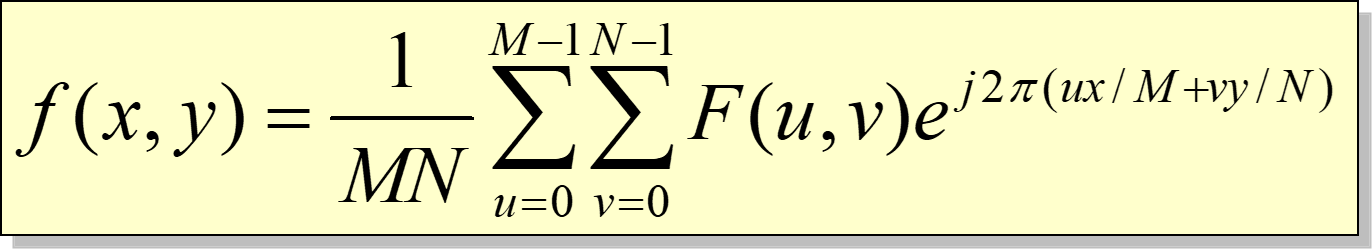
**Fourier Series** tells us that any function can be represented as a sum of sines/cosines of different frequencies multiplied by a different coefficient. Similarly, non periodic functions can also be represented as the integral of sines/cosines multiplied by weighing function.

The Discrete Fourier Transform of f(x, y), for x = 0, 1, 2…M-1 and y = 0,1,2…N-1, denoted by F(u, v)



for u = 0, 1, 2…M-1 and v = 0, 1, 2…N-1.

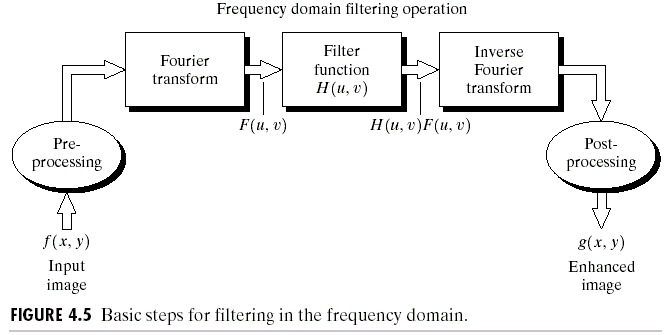
It is really important to note that the Fourier transform is completely reversible .The inverse DFT is given by:

****

for *x* = 0, 1, 2…M-1 and *y* = 0, 1, 2…N-1

The filtering in frequency domain consists of following steps:

1. Compute F(u,v) the DFT of the image
2. Multiply F(u,v) by a filter function H(u,v)
3. Compute the inverse DFT of the result

****

The following steps can be followed to process an image in frequency domain:

1. Take Fourier Transform of the image
2. Shift its Dc component to center
3. Obtain absolute values of the output
4. Brighten up the image using log transformation
5. Normalize the image to obtain High Contrast (min, max) ->(0, 255)

## Some Useful Commands:

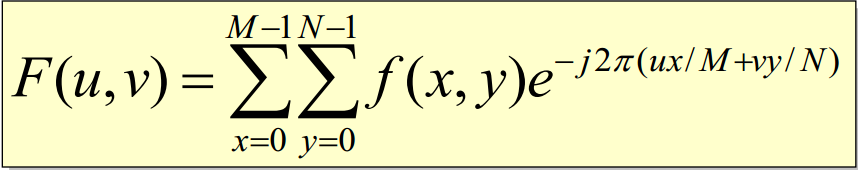
1. To obtain the Fourier Transform of an image: my\_transformed\_image = **numpy.fft.fft2(my\_image)**
2. To obtain the Inverse Fourier Transform of an image: my\_inverse\_image = **numpy.fft.ifft2(my\_image)**
3. To shift the DC component of a Fourier Transformed Image to center: my\_shifted\_image = **numpy.fft.fftshift(my\_transformed\_image)**
4. To shift the DC component back to the top left corner: my\_inverse\_shifted\_image = **np.fft.ifftshift(my\_shifted\_image)**
5. To calculate absolute of a value: my\_absolute = **numpy.abs(my\_image)**
6. To calculate the exponential of an element: my\_exponential = **numpy.exp(my\_input)**
7. To use the value of pi: **numpy.pi**
8. To denote a complex number: **simply put j after it** e.g. **-1j**
9. To multiply two matrices point by point: my\_result = **numpy.multiply(my\_image, my\_filter)**
10. To take log transformation of the image: my\_result = numpy.log(my\_image)
11. To use math function: import math, x = math.sqrt(25)
12. To normalize an image: my\_image\_normalized = **cv2.normalize(my\_source\_image, my\_desitnation\_image, new\_min\_value, new\_max\_value, NORM\_MINMAX, CV\_8UC1)**

The argument NORM\_MINMAX tells the function which method to use to normalize the image while the argument CV\_8UC1 tells the number of channels that the destination image will have. In other words, CV\_8UC1 means a grayscale image in unit8.

## Lab Tasks:

**Lab Task 1:**

The discrete Fourier Transform of an image can be calculated as follows:



Use the image Fig01.tif to compute the DFT.

The following transformation can be applied to the images (so that the Fourier transform the spectrum will be shifted at the R/2, C/2 location).

**F(x, y) = F(x, y)\*(-1) ^ (x + y)**

Now, repeat the above using the built in functions and display the results of both.

**Lab Task 2:**

Create an ideal low pass and high pass filter using the distance map. The size of the distance map should be equal to the size of the image. Take DFT of the Fig01.tif and dot multiply the transformed image and the filter. Take inverse DFT of the image and display the results.

**Lab Task 3:**

Filters that are created in spatial domain can be applied in frequency domain. In order to do that, DFT of both the image and filter is to be taken, dot multiplied in the frequency domain and then converted back to spatial domain. Before converting both the image and the filter in frequency domain, their size has to be made the same. This can be done by making the size of both the image and filter equal to:

(Image size + Filter Size) – 1

And placing the image and the filter in the top left corner.  
Create a Gaussian filter of 9x9 in spatial domain. Then after making sure that the size of the image and the filter is the same (in accordance to the above formula) convert them both into frequency domain. Dot multiply the image and the filter in frequency domain and then convert them back to spatial domain to check the effects.  
Also display the Gaussian filter in frequency domain.

## Home Task:

Utilize frequency-domain techniques to identify and remove periodic noise from an image, ensuring a clean and visually improved output using a notch filter. Take any image that has *periodic* noise and apply this technique to remove it.

## Conclusion:

This lab has given an introduction of Fourier transform for image processing and application of different filters for image enhancement.